Designing Equitable Data Center Scheduling Systems

Sahana Rangarajan, Xuesi Chen, Pratyush Patel*, Jaylen Wang, Akshitha Sriraman

Carnegie Mellon University, University of Washington*
Email IDs: ssrangar@andrew.cmu.edu, xuesic@andrew.cmu.edu, patelp1@cs.washington.edu, jaylenw@andrew.cmu.edu, akshitha@cmu.edu

I. Research problem

Today, humans rely on large-scale computing infrastructures, i.e., data centers (DCs), to satisfy their computing needs [1]. For example, six in ten people use modern web services such as social media, online messaging, web search, video streaming, and online banking that require DCs that scale to hundreds of thousands of high-end computers or servers [1], [2].

Large-scale DC computing is fundamentally inequitable in several ways due to its distributed implementation that often prioritizes tasks [3]. For example, prior work has found that web search result quality varies across diverse end-users, with certain results prioritized or hidden based on user profiles [4]. Hence, it is critical to identify when prioritization-based solutions in modern DC web systems can result in biases, compromising equity. To the best of our knowledge, we are the first to explore equity as a first-order hardware/software system design metric.

In this work, we investigate whether prioritization-driven algorithmic bias in DC scheduling systems (especially those that are ML-driven) can make inequitable scheduling decisions to improve performance. Specifically, we posit that bias may stem from user demographics. For example, to improve latency headroom, a scheduler might supply a slower response to a user in a remote location with poor internet connection (vs. a user in an urban area). Similarly, a scheduler might be able to exploit the notion that an older user might be more patient and willing to wait longer for a response [5], causing active discrimination against users.

Designing equitable scheduling systems is especially important since several modern web applications can be life critical. For example, in the 1995 Kobe Earthquake, initial earthquake updates and information on relief resources were disseminated via internet-scale web services in the absence of operational physical phone lines [6]. More recently, social media has become essential to emergency response, both in mitigating primary deaths (resulting directly from the emergency) and secondary deaths (resulting from infrastructure breakdown) [7]. Today, during and after the initial impact of disasters and emergency situations, first responders and survivors use web services to communicate between each other and among themselves. Hence, making unbiased DC scheduling decisions is essential to ensure reliable and rapid web service response in these situations.

II. Background and Related Work

Due to the volume of connections and computing occurring concurrently in large-scale web systems, workload distribution and task prioritization over limited DC resources is an important component in determining Quality of Service (QoS) [8]. To this end, current schedulers primarily prioritize efficiency metrics such as performance-, cost- [9]–[11], and energy-efficiency [12].

Many scheduler designs incorporate Machine Learning (ML) models (e.g., deep reinforcement learning [13], reinforcement learning, and neural networks [14]). However, as shown by prior works [15] [16], the use of learning algorithms may cause algorithmic bias. One notorious example is COMPAS, an AI-based criminal justice tool that was found to predict higher recidivism rates for Black defendants [17]. Although the context is different, such biases could also arise in ML-driven DC scheduling algorithms if not monitored, creating an unintentional layer of inequity in web systems beyond those already known and studied. Typically, modern DC schedulers aim to optimize resources utilization based on the QoS requirements (e.g., 300 ms response latency [18]) of each application or user. Hence, if discrete demographics of web service users represent certain groups of traits and constraints that schedulers optimize by (e.g., tolerance to a delayed response), the schedulers could "learn" bias unintentionally.

Another side to the question of scheduler bias is whether there is an ethical way for schedulers to deliberately leverage such biases. If we find demographic-based QoS tolerances through the user study, part of the task of monitoring schedulers for bias would also be to determine whether learning schedulers already exploit such correlations.

III. Key Contributions

The first part of our work is to determine whether users' perceptions of web service latency are impacted by their demographics. To further investigate this question, we are designing a user study composed of three phases: a pre-task demographic questionnaire, search tasks on a custom web application, and a post-task response survey. The questionnaire and survey will provide the data to analyze whether tolerance to web service response delays has any bearing on respondent demographics.

Driven by this study's results, we will next design a scheduler that deliberately leverages demographic-dependent latency tolerance to suitably make inequitable scheduling decisions. Using this scheduler, we will determine the performance improvement that a DC scheduler might achieve by trading off equity for latency.

Finally, we will create a scheduling framework that monitors and corrects unintentional algorithmic biases under the hood. To measure demographic-based algorithmic bias, we propose monitoring latency per user-type, where user-type is a multi-dimensional feature vector composed of demographic information. Our framework will include real-time bias assessments, so that the deployed scheduler can determine the existence of bias and address it immediately.

There are currently two main methods of evaluating algorithmic bias—(1) algorithmic impact assessments, which analyze concrete societal impacts that arise through an algorithm-based system, and (2) auditing, which evaluates the underlying technology itself for bias metrics [19]. Since the former involves risk and impact assessments rather than a technical evaluation of the scheduling algorithms as we propose, our current solution will fall under auditing. However, impact assessments could be a logical further step once auditing is in place.

IV. Preliminary Experiments

For the experimental design of our user study, we draw from the Internet latency studies performed by Borella et al. [5] and Arapakis et al. [20]. We are creating a web application to run on a desktop browser, structured as a simple wiki based on the Bookstack open source project [21]. Here, we include a homepage with five options for navigation, each of which takes participants to a page with information pertinent to several predefined search tasks, drawn from Kelly et al. [22]. Fig. 1 depicts a basic UI mock of the web application based on Bookstack's design.

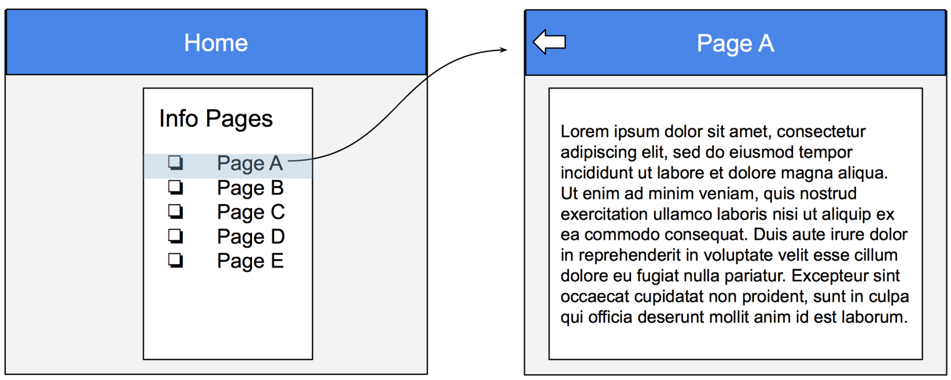

Fig. 1: User Study Web Application UI Mock

Without the participants' knowledge, we also vary the render time of the info pages to imitate varying latencies. We then poll participants' levels of frustration or satisfaction with the web application, and use these reactions to determine whether there are notable correlations to draw between user demographic profiles, latencies, and user satisfaction (i.e., correlating QoS with user demographics). So far, from reviewing existing literature around the relation between latency tolerance and demographics, we have found the factors of age and spoken language to be potential features of interest, which we will particularly survey for [5]. For both the pre-task demographic questionnaire and the post-task survey, we are utilizing the Questionnaire for User Interaction Satisfaction, or QUIS, developed by UMD [23].

Next, to create an intentionally biased scheduling system, we will build a scheduler based on an existing ML-driven scheduler, such as Paragon [24], Quasar [8], Decima [14], or DeepRM [13]. We will then augment or modify the datasets from which the schedulers learn based on the results of our user study. Essentially, we will intentionally push a scheduler towards a bias based on the demographic features we observed, measuring the ensuing performance improvements.

Finally, we will build our bias-free scheduler framework. The user features our auditing framework monitors will be the same demographics we initially used to bias the scheduler. For example, the auditing tool could collect data on user ages and the latency they experience and raise an alert if the scheduler routinely deprioritizes requests from older users. Additionally, various tools currently exist to measure a breakdown of network latency and performance, such as Performance-Bookmarklet [25] and Page Load Time [26]. We could leverage these tools to more accurately determine for what part of the end-to-end latency the scheduler is responsible.

References

[1] Akshitha Sriraman, Abhishek Dhanotia, and Thomas F Wenisch. SoftSKU: Optimizing Server Architectures for Microservice Diversity @Scale. In *The International Symposium on Computer Architecture*, 2019.

[2] 6 in 10 people around the world now use the internet. https://datareportal.com/reports/6-in-10-people-around-the-world-now-use-the-internet.

[3] S Derek Turner. *Digital Denied: The Impact of Systemic Racial Discrimination on Home-Internet Adoption*. Free Press, Dec 2016.

[4] Xinyu Xing, Wei Meng, Dan Doozan, Nick Feamster, Wenke Lee, and Alex C. Snoeren. Exposing inconsistent web search results with bobble. *Passive and Active Measurement*, page 131–140, 2014.

[5] M.S. Borella, A. Sears, and J.A. Jacko. The effects of internet latency on user perception of information content. In *GLOBECOM 97. IEEE Global Telecommunications Conference. Conference Record*, volume 3, pages 1932–1936 vol.3, 1997.

[6] D.E. Newsom, C.L. Herzenberg, and C.E. Swieltik. Value of the internet in emergency response. *IPCC 99. Communication Jazz: Improvising the New International Communication Culture. Proceedings 1999 IEEE International Professional Communication Conference (Cat. No.99CH37023)*, 1999.

[7] Perumalraja Rengaraju, Kamalesh Sethuramalingam, and Chung-Horng Lung. Providing internet access for post-disaster communications using balloon networks. *Proceedings of the 18th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, amp; Ubiquitous Networks*, 2021.

[8] Christina Delimitrou and Christos Kozyrakis. Quasar. *ACM SIGPLAN Notices*, 49(4):127–144, 2014.

[9] Alexey Tumanov, Timothy Zhu, Jun Woo Park, Michael A. Kozuch, Mor Harchol-Balter, and Gregory R. Ganger. Tetrisched. *Proceedings of the Eleventh European Conference on Computer Systems*, 2016.

[10] Ionel Gog, Malte Schwarzkopf, Adam Gleave, Robert N. M. Watson, and Steven Hand. Firmament: Fast, centralized cluster scheduling at scale. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 99–115, Savannah, GA, November 2016. USENIX Association.

[11] Robert Grandl, Srikanth Kandula, Sriram Rao, Aditya Akella, and Janardhan Kulkarni. GRAPHENE: Packing and Dependency-Aware scheduling for Data-Parallel clusters. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 81–97, Savannah, GA, November 2016. USENIX Association.

[12] David Lo, Liqun Cheng, Rama Govindaraju, Luiz André Barroso, and Christos Kozyrakis. Towards energy proportionality for large-scale latency-critical workloads. In *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, pages 301–312, 2014.

[13] Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. Resource management with deep reinforcement learning. In *HotNets-XV, Atlanta, GA, USA*, November 2016.

[14] Hongzi Mao, Malte Schwarzkopf, Shaileshh Bojja Venkatakrishnan, Zili Meng, and Mohammad Alizadeh. Learning scheduling algorithms for data processing clusters. *Proceedings of the ACM Special Interest Group on Data Communication*, 2019.

[15] Jennifer W. Tsai, Jessica P. Cerdeña, William C. Goedel, William S. Asch, Vanessa Grubbs, Mallika L. Mendu, and Jay S. Kaufman. Evaluating the impact and rationale of race-specific estimations of kidney function: Estimations from u.s. nhanes, 2015-2018. *eClinicalMedicine*, 42:101197, 2021.

[16] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Computing Surveys*, 54(6):1–35, 2021.

[17] Julia Dressel and Hany Farid. The accuracy, fairness, and limits of predicting recidivism. *Science Advances*, 4(1), 2018.

[18] Akshitha Sriraman and Thomas F Wenisch. μTune: Auto-Tuned Threading for OLDI Microservices. In *Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation*, 2018.

[19] Emre Kazim and Adriano Soares Koshiyama. A high-level overview of ai ethics. *Patterns*, 2(9):100314, 2021.

[20] Ioannis Arapakis, Souneil Park, and Martin Pielot. Impact of response latency on user behaviour in mobile web search. *Proceedings of the 2021 Conference on Human Information Interaction and Retrieval*, 2021.

[21] BookStackApp. Bookstackapp/bookstack: A platform to create documentation/wiki content built with php amp; laravel, 2022.

[22] Diane Kelly, Jaime Arguello, Ashlee Edwards, and Wan-ching Wu. Development and evaluation of search tasks for iir experiments using a cognitive complexity framework. *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, 2015.

[23] Quis™ - the questionnaire for user interaction satisfaction, Mar 1999.

[24] Christina Delimitrou and Christos Kozyrakis. Paragon. *Proceedings of the eighteenth international conference on Architectural support for programming languages and operating systems - ASPLOS '13*, 2013.

[25] Micmro. Micmro/performance-bookmarklet: Performance-bookmarklet helps to analyze the current page through the resource timing api, navigation timing api and user-timing - requests by type, domain, load times, marks and more. sort of a light live webpagetest.

[26] Alex-Vv. Alex-vv/page-load-time: Google chrome extension to measure page load time and display it in the toolbar.